Passport Challenge Project (Alexey Kalinin)

Requirements

- The tree should contain a group of nodes, with a main (root) node that can have any number of 'factories'.
- These factory nodes can in turn generate a set amount of random numbers (up to15), represented as child nodes of their respective factories.
- Factories and children should be created through some means of user input (right click, button press, etc) specifying the number of children to generate (up to 15)and the ranges of those children.
- Factories should have an adjustable name assigned to them, be removable, and have an adjustable lower and upper bound for the random number generation.
- You may use any programming languages and front-end design styles of your choosing to create the project.
- All users should see any changes made to the tree immediately across browsers without refreshing or polling.
- The state of the tree should remain persistent; reloading should not undo any state.
- All of a factory's existing child nodes should be removed upon each new generation.
- Your project should be secure, validate inputs, and protect against injections.
- Your project should be hosted on the web using a service such as Amazon AWS or Heroku to run your submission.
- The project should exhibit both a front-end and back-end codebase built by you. Please submit your project, link, and source to the email listed below

Assumptions

1. Factories and nodes contains in-sensitive data, so no authentication is provided to access the app. It is publicly hosted and can be accessed by anyone without any special privileges.
2. Due to the nature of the app listed in the previous bullet point, all the REST API calls are un-authenticated.

Technology Stack

1. Laravel 5.1 (PHP 7.0) for main application functionality and to access API

2. NodeJS and Express module for building REST API
3. MongoDB – NoSQL database to store the data
4. Bootstrap, JQeury, Bootstrap Treeview Plugin - to build GUI
5. Redis (through Socket.IO) to build publisher/subscriber mechanism for real-time data updates

Hosting

1. Laravel application is hosted on Heroku: http://passport-factories-web.herokuapp.com/
2. REST API is hosted on Heroku: https://passport-factories-api.herokuapp.com/

3. Redis pub/sub server is also hosted on Heroku: https://passport-factories-redis.herokuapp.com/
4. MongoDB is hosted on AWS EC2 instance on a FreeTier with Ubuntu OS: ec2-34-224-23-245.compute-1.amazonaws.com

GIT links

https://github.com/alkaline184/passport-api
https://github.com/alkaline184/passport-redis
https://github.com/alkaline184/passport-factories

Architecture

Application is heavily relied upon the NodeJS Express server, which serves REST API for consumption. Node JS connects to the MongoDB database and stores all the information in the plain Document structure.

Factories collection:
_id: ObjectId("595281834fae8a0004f312da")
Name: Hello World!
Lower: 2
Upper: 13
Count: 3
Children: [3,5,7]

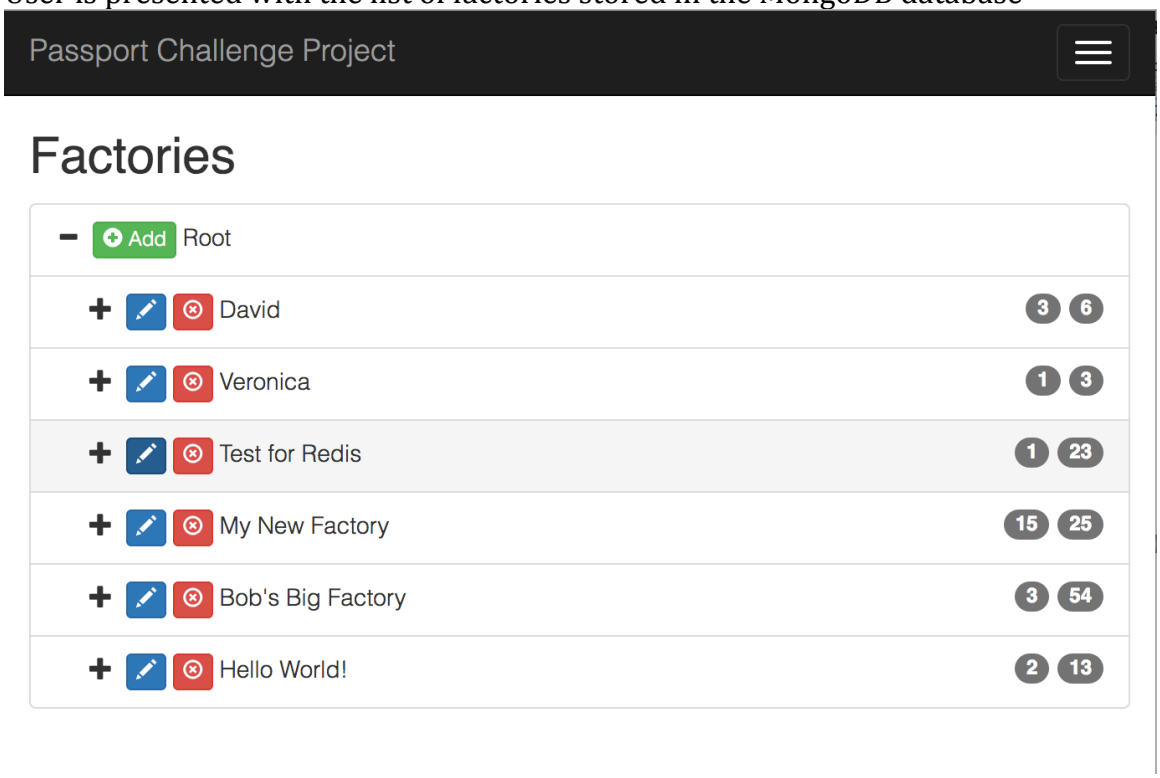
This is a simple tree with defined number of levels (2), so we don't need to store any parent/child relationships in the Relational DB's, that's why MongoDB document structure is sufficient.

Laravel application consumes REST API and serves the views based on the JSON data received from the REST calls. It also works as a publisher for Redis pub/sub module. Every time when the Factory gets created/updated/deleted, its publishes an event to Redis server. The main layout also has a JavaScript Redis subscriber that is being notified of every single change and updates the TreeView component with the newest data.

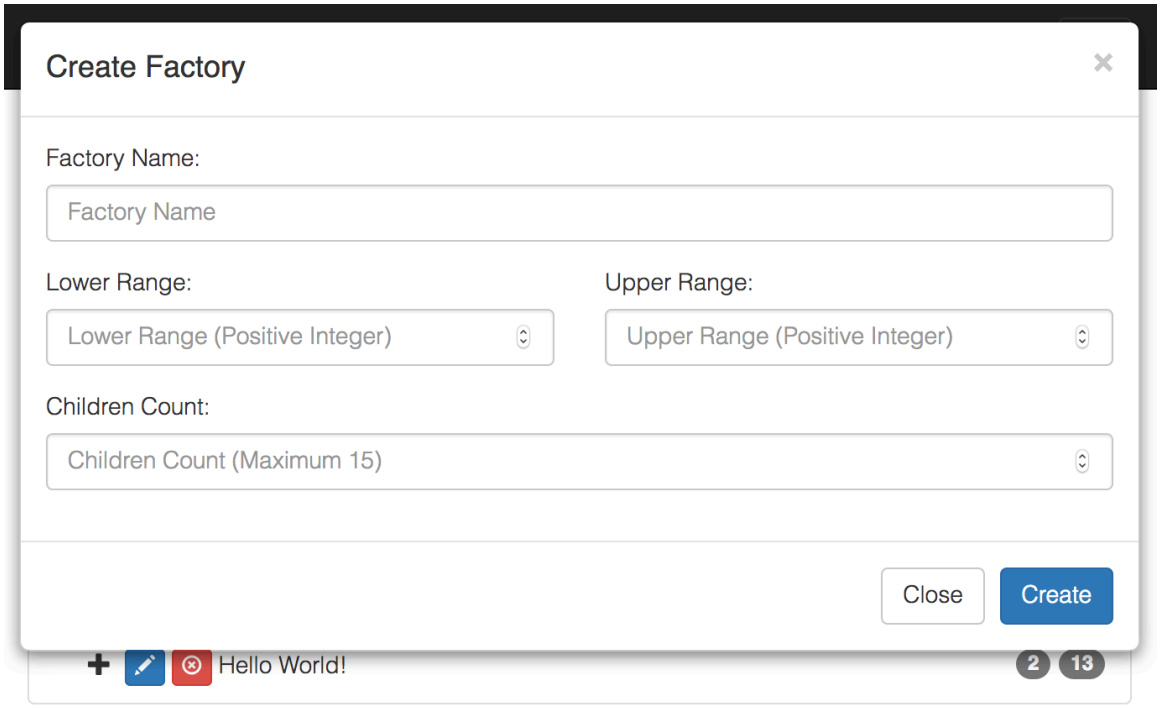
The frontend of the application is built on top of the Bootstrap library that provides responsive layout.

Use Cases

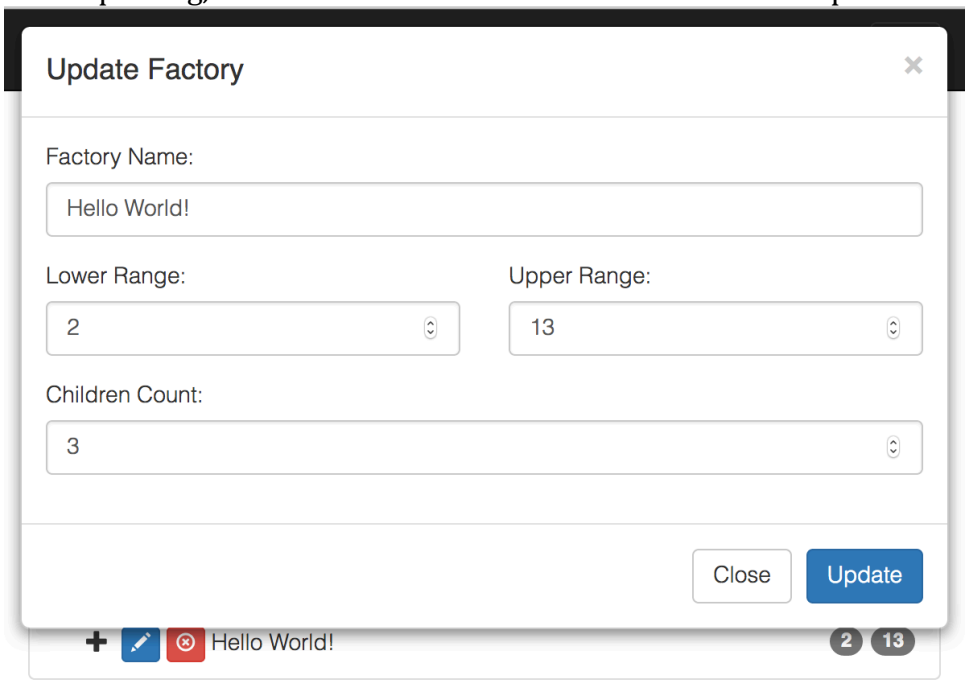
1. User is presented with the list of factories stored in the MongoDB database



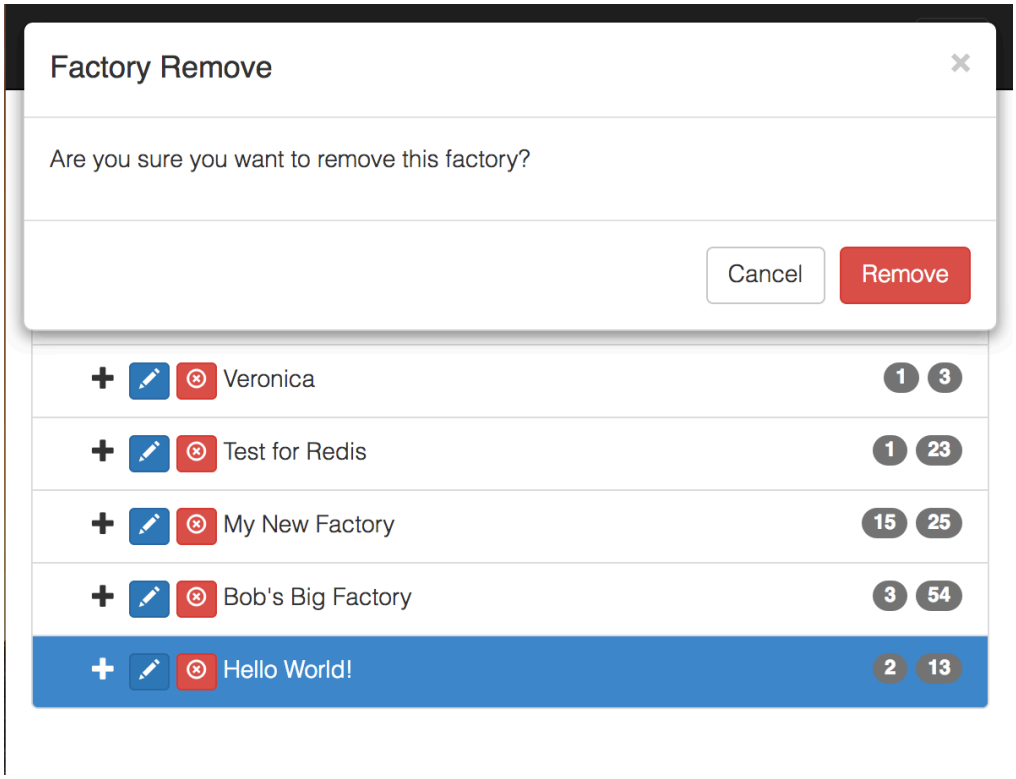
2. User can click on “Add” button, which will open up a dialog box to create a new factory. It provides the client-side validation for all the input fields:
   Name: Not Blank
   Lower: Positive Integer
   Upper: Positive Integer
   Count: Positive Integer not greater than 15
   Lower should be less than Upper.
   REST API provides the same validation as well and returns proper errors codes for Bad Requests (400).



3. As soon as User clicks on "Create" button, application makes a Request to REST API and saves the new Factory into the Database. It also automatically makes another call to update children of this factory based on lower, upper and count variables. After that it makes an automatic call to refresh the full tree. It also publishes a message to Redis at the same time.
4. User can update any factory and any variable of this factory by clicking on "Pencil" button next to each factory name. It will pull up the current information for this factory and user will be able to modify and re-save it. After updating, it will re-add new children based on the new parameters.



5. User can also delete any factory by clicking on "Cross" button next to each factory. It will show the prompt dialog box to confirm the delete action.



REST API

Create new factory

Method: POST
URL: https://passport-factories-api.herokuapp.com/factories/
Request body:

```
{
  "name": "New Factory",
  "lower": "12",
  "upper": "34",
  "count": "9"
}
```

Successful Response:

```
{
  "name": "New Factory",
  "lower": "12",
  "upper": "34",
  "count": "9",
  "children": "[]",
  "_id": "595292b98a60f5000429d0aa"
}
```

Error response: 400 Bad Request

```
{
  "Bad Request": "Upper Range should be a Positive Integer"
}
```

Update factory

Method: PUT
URL: https://passport-factories-api.herokuapp.com/factories/%5Bfactory_id%5D
Request body:

```
{
   "name": "New Factory name",
   "lower": "12",
   "upper": "34",
   "count": "9"
}
```

Successful Response:

```
{
   "name": "New Factory",
   "lower": "12",
   "upper": "34",
   "count": "9",
   "children": "[]"
}
```

Error response: 400 Bad Request

```
{
   "Bad Request": "Specified factory doesn't exist"
}
```

Delete factory

Method: DELETE
URL: https://passport-factories-api.herokuapp.com/factories/%5Bfactory_id%5D

Successful Response:
Factory 595292b98a60f5000429d0aa deleted

```
{
   "Bad Request": "Specified factory doesn't exist"
}
```

Get Factory

Method: GET
URL: https://passport-factories-api.herokuapp.com/factories/%5Bfactory_id%5D

Successful Response:

```
{
   "_id": "595281834fae8a0004f312da",
   "name": "Hello World!",
   "lower": "2",
   "upper": "13",
   "count": "3",
   "children": [
```

```
      2,
      12,
      10
    ]
}
```

Error response: 400 Bad Request

```
{
   "Bad Request": "Specified factory doesn't exist"
}
```


Get All Factories

Method: GET
URL: https://passport-factories-api.herokuapp.com/factories/

Successful Response:

```
[
  {
    "_id": "59501d0fa4e06a0ea3ff6a46",
    "name": "David",
    "lower": "3",
    "upper": "6",
    "count": "5",
    "children": [
      5,
      3,
      4,
      4,
      5
    ]
  },
  {
    "_id": "59501d55a4e06a0ea3ff6a4a",
    "name": "Veronica",
    "lower": "1",
    "upper": "3",
    "count": "1",
    "children": [
      2
    ]
  },
{
    "_id": "5951ce736db2310004237dd5",
    "name": "My New Factory",
    "lower": "15",
    "upper": "25",
    "count": "2",
    "children": [
      17,
      18
    ]
```

```
  },
  {
    "_id": "595281834fae8a0004f312da",
    "name": "Hello World!",
    "lower": "2",
    "upper": "13",
    "count": "3",
    "children": [
      2,
      12,
      10
    ]
  }
]
```


Add Children

Method: PUT
URL: https://passport-factories-api.herokuapp.com/factories/%5Bfactory_id%5D/children
Successful Response:


```
{
  "_id": "595281834fae8a0004f312da",
  "name": "Hello World!",
  "lower": "2",
  "upper": "13",
  "count": "3",
  "children": [
    10,
    5,
    11
  ]
}
```


Error response: 400 Bad Request


```
{
  "Bad Request": "Specified factory doesn't exist"
}
```